

**Informatics Institute of Technology**

**Department of Computing**

**5SENG003C Algorithms Theory Design and practice**

# Task 5

**Module : 5SENG003C Algorithms**

**Student First Name : Pramuditha**

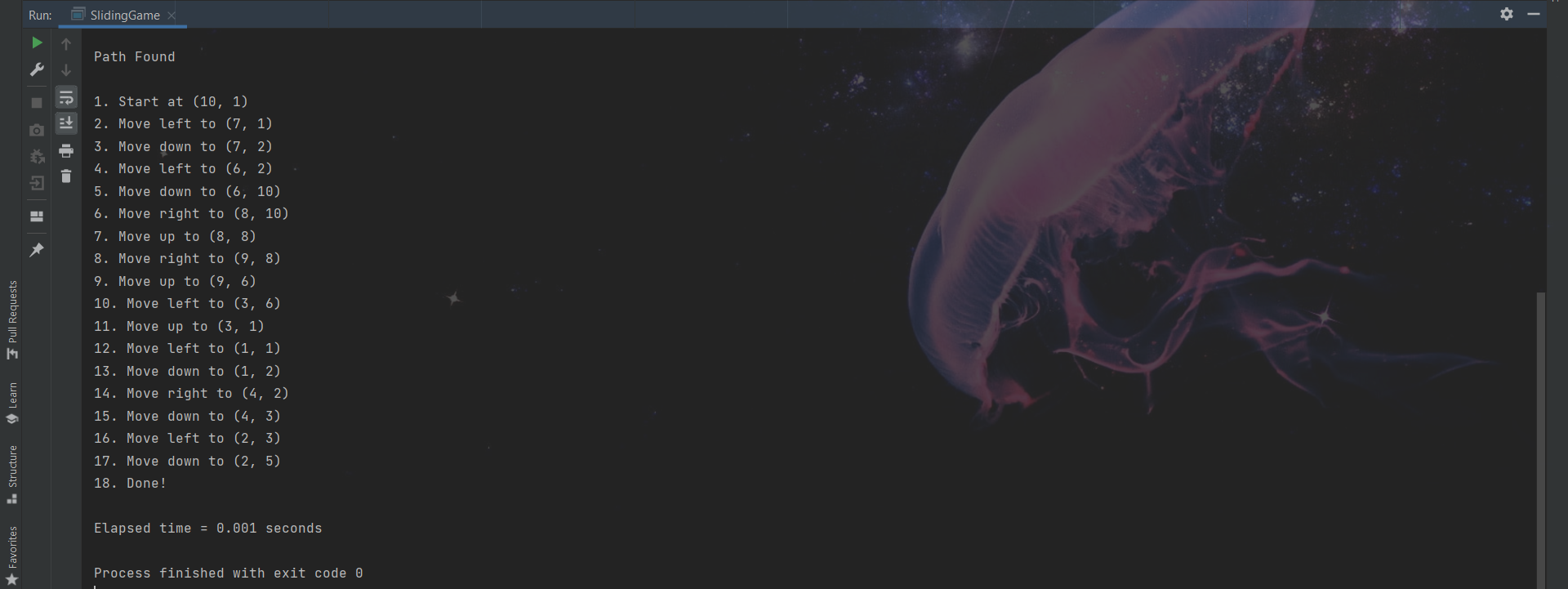
**Student Surname : Karunarathna**

**Student ID : 20200612 / w1836048**

1. There are many algorithms to find the path among nodes in graphs. Path finding algorithms such as A\* and Dijkstra, as well as Depth-first search and Breadth first search are some algorithms that can be used to find the paths among the nodes. Among those algorithms this program uses the Breadth-first search. Breadth-first search is an algorithm that can be used to traverse a graph to satisfy certain conditions. This algorithm travels each and every node in the graph by selecting a single node and its neighboring node. Breadth-first search traverse through nodes in breadth ward motion while Depth-first search traverse in depth ward motion. Accordingly, this algorithm can be used to search for the shortest path among the nodes. BFS uses a queue like data structure to find the shortest path. Depth-first search is more suitable when the target is away from the source while a breadth-first search is more suitable when the target is closer to the source. The time complexity of BFS is O(V+E) where E is the edges and V is the vertices. Also, BFS can traverse through nodes with the smallest number of iterations. Due to the simplicity of the architecture and high-level accuracy of the results when compared to the other algorithms this program uses the Breadth-first search algorithm to find the shortest path while sliding the node.

To store the map as a grid the program has used a character two-dimensional array. With the use of a 2D array, it is easy to get the location of each and every node in the graph accurately. In the same manner, this program used a Boolean 2D array to store the nodes in the correct location that were already traversed. Also instead of a queue, this program has used an array list due to the simplicity of implementation and a reduction of the code complexity. In order to save the path this program use an array list instead of a queue because in queue it retrieves elements in first come first out basis. According to the program it should retrieve in last come first out basis. The number of paths that stored in this program is not constant. Accordingly, this program uses an array list instead of an array and instead of a queue to store the paths that retrieve and print to the console.

1. Graphical user interface, text

   Description automatically generated

c)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Time Consume (in Seconds)** | | | | |  |  |
| **Index** | **Input Puzzle** | **1st Run** | **2nd Run** | **3rd Run** | **4th Run** | **5th Run** | **Average Time** | **Ratio** |
| 1 | puzzle\_20.txt | 0.001 | 0.001 | 0.002 | 0.001 | 0.001 | 0.0012 | 2 |
| 2 | puzzle\_40.txt | 0.001 | 0.002 | 0.003 | 0.002 | 0.004 | 0.0024 | 1.25 |
| 3 | puzzle\_80.txt | 0.003 | 0.005 | 0.003 | 0.002 | 0.002 | 0.003 | 1.333333333 |
| 4 | puzzle\_160.txt | 0.003 | 0.005 | 0.004 | 0.005 | 0.003 | 0.004 | 1.95 |
| 5 | puzzle\_320.txt | 0.006 | 0.008 | 0.009 | 0.007 | 0.009 | 0.0078 | 1.897435897 |
| 6 | puzzle\_640.txt | 0.014 | 0.014 | 0.017 | 0.014 | 0.015 | 0.0148 | 2.202702703 |
| 7 | puzzle\_1280.txt | 0.031 | 0.031 | 0.032 | 0.036 | 0.033 | 0.0326 | 2.355828221 |
| 8 | puzzle\_2560.txt | 0.08 | 0.073 | 0.071 | 0.083 | 0.077 | 0.0768 | 0 |